MODELING AND VISUALIZATION OF VERSION-CONTROLLED DOCUMENTS

A Thesis
Presented to
The Academic Faculty

by

Seungyeon Kim

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in the
School of Computer Science

Georgia Institute of Technology
May 2011

MODELING AND VISUALIZATION OF VERSION-CONTROLLED DOCUMENTS

Approved by:

Professor Guy Lebanon, Advisor
School of Computer Science
Georgia Institute of Technology

Professor Alexander Gray
School of Computer Science
Georgia Institute of Technology

Professor John Stasko
School of Computer Science
Georgia Institute of Technology

Date Approved: 29 March 2011

ACKNOWLEDGEMENTS

First and foremost, I offer my sincere thanks to Professor Guy Lebanon, who has guided me throughout my Master's program with his expertise and patience. He taught me to think outside the box and encouraged me to challenge uncharted areas, which I believe was essential for the success of my research. From him, I also learned to balance the theoretical research and practical applications. I also thank Professor Alex Gray and John Stasko for being on my thesis committee, giving me invaluable input and guiding my research.

I would also like to thank Joshua V. Dillon who co-authored many of the publications resulting from this work: for amazing ideas and help in writing papers. Without him, the second part of the paper would have not been completed. He has a different approach to developing ideas, which always resulted in fruitful discussions. I would like to thank all Statistical Machine Learning and Visualization Lab members: Mingxuan Sun, Krishnakumar Balasubramanian, Joonseok Lee, Fuxin Li and Yi Mao. For my lab mates in 1305, thank you for all the fun and interesting discussions - they were always an inspiration for my research.

I'd like to thank for all of my friends. Thanks for my friends in Georgia Tech for all the support, especially for free coffees (though, I think I ended up paying more often). Thanks for my friends in U.S. for sharing tips on enjoying life as a graduate student. I should also mention my old friends who give me great warmth. I am very happy that we kept in touch, especially from Seoul National University, ALC and literature club (Breath) members.

Finally, I would like to thank my family; especially my grandparents and parents for their endless love and support, and my sister for tips on my fashion.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS OR ABBREVIATIONS

CRM Cumulative Rversion Map.

CVS Concurrent Versions System.

LCS Longest Common Subsequence.

LSS Local Space-time Smoothing.

SVN Subversion.

SUMMARY

Version-controlled documents, such as Wikipedia or program codes in Subversion, demands a novel methodology to be analyzed efficiently. The documents are continually edited by one or more authors in contrast of the case of static documents. These collaborative processes make traditional methodologies to be ineffective, yet needs for efficient methodologies are rapidly developing.

This thesis proposes two new models based on *Local Space-time Smoothing*(LSS) which captures important revision patterns while *Cumulative Revision Map*(CRM) tracks word insertions and deletions in particular positions of a document. These two methods enable us to understand and visualize the revision patterns intuitively and efficiently. Synthetic data and real-world data are used to demonstrate its applicability.

CHAPTER I

INTRODUCTION

Wikipedia is a model example of a version-controlled document. The Wiki system, CVS, and SVN are popular systems that help collaborative authoring while automatically provide contents of revisions, the submission time, and the author. The version-controlled document system is commonly used in building a book or a large computer code project. With the huge success of Wikipedia project, version-controlling gained the public spotlight, and now the number of version-controlled documents is rapidly increasing.

Automatic analyzing version-controlled documents is in great demand. In the past, only a small group of people were needed to meticulously review a draft of a book to polish it; however, a widely and actively edited document, Wikipedia for example, cannot be handled in that way today. Millions of users are simultaneously editing materials on billions of topics. Without an automatic analysis on version-controlled documents, the quality of collaborative writing will remain in disappointing.

In this thesis, two approaches are taken to model and visualize version-controlled documents. *Local Space-time Smoothing*(LSS) and *Cumulative Revision Map*(CRM). LSS focuses on providing probabilistic model of version-controlled documents in Chapter 3, and CRM is a new information visualization technique for version-controlled documents in Chapter 4.

LSS is a continuous representation of a version-controlled document in probabilistic interpretation, which fulfills the need of mathematical model for the automatic analysis of version-controlled document. There is still not an effective model for version-controlled documents even though there are hundreds of traditional document

models since modeling a document is a major issue in computational linguistics. The traditional ways to model a document regarded a document as a sequence of words, in contrast version-controlled documents consist of a history of sequences of words during the authoring process. The revisions, as the differences between consecutive versions, may be authored by a single author or by multiple authors working collaboratively. Not only the modeling of the final word sequence but also the modeling the entire history of sequences is required for a complete model.

LSS generalize the *locally weighted bag of words* representation [22]. The representation smooths the sequence of version-controlled documents across two axis, time and space. The time axis represents each revisions and the space axis represents document position, starting from the beginning of a text to the end of it. The smoothing results in a continuous map from a space-time domain to the simplex of term frequency vectors.

$$\gamma : \Omega \to \mathbb{P}_V \quad \text{where} \quad \Omega \subset \mathbb{R}^2, \quad \mathbb{P}_V = \left\{ w \in \mathbb{R}^{|V|} : w_i \geq 0, \ \sum_{i=1}^{|V|} w_i = 1 \right\} \tag{1}$$

Equation (1) captures the variation in the local distribution of word content across time and space. Thus $[\gamma(s,t)]_w$ is the (smoothed) probability of observing word w in space s (document position) and time t (version). Geometrically, γ realizes a divergence-free vector field (since $\sum_w [\gamma(s,t)]_w = 1$, γ has zero divergence) over the space-time domain Ω.

LSS is evaluated with four tasks: visualizing content changes in space and time, detecting noticeable transitions in word content, segmenting the space-time domain, and predicting future editing operations. Each of them is sub-tasks of reviewing process of collaborative authoring and provides a useful insight for the editing pattern of the document. Each task is depending on the values, $\gamma(s,t)$ and its derivation $\dot{\gamma}(s,t)$. Synthetic data and real-world data were used in the four tasks. The synthetic

data is a sequence of words that randomly picked from several probability settings. It proves the concept of the model. Real-word version-controlled documents were collected in Wikipedia and Google Wave. The results of the real-world data shows will show the usefulness of the model.

The Cumulative Revision Map(CRM) focuses more on information visualization and provides an intuitive representation of the additions and deletions of contents throughout the revision of a version-controlled document. While a traditional technique, *Unix diff*, solves *Longest Common Subsequence problem* and generates an edit script between only two revisions, CRM maintains the entire addition and deletion history of a document.

CRM uses a graph structure with a node containing a subsequence of a document. In each revision of the document, CRM solves the LCS problem in the same way as *Unix diff* between previous and current revision. CRM preserves the unchanged part of the document intact, while splitting a node to add new contents and to delete respectively. In the most recent version of a document, CRM has a complete history of revisions of a document visible as a map.

CRM visualize an entire history as a map, which is a scalable and interactive interface. The vertical position of a node represents revision and the horizontal position show the position of a word in a document. Nodes and edges can be simplified to provide a scaleable representation of a version-controlled document. Edges can be changed to vertical lines while shrinking all horizontal gaps between nodes. This simplification approach also gives a more concise document history layout. Moreover, a user can easily pinpoint a portion on the map to obtain valuable information because it is a map that can be intuitively understood.

Chapter 3 is based on the work published on COLING 2010 [21] and Chapter 4 is a joint work with Joshua V. Dillon and it is submitted on Infovis 2011.

CHAPTER II

RELATED WORK

2.1 Probabilistic Document Modeling

2.1.1 Local Smoothing Approach

Although probabilistic document modeling is an actively researched area, there has been a few works of modeling version-controlled documents. However, *Local Space-time Smoothing* is still founded on the traditional probabilistic document modeling, and extend it maintaining theoretical support from traditional document modelings.

Our approach is the first to consider version-controlled documents as continuous mappings from a space-time domain to the space of local word distributions. It extends the ideas in [22] of using kernel smoothing to create a continuous representation of documents. In fact, our framework generalizes [22] as it reverts to it in the case of a single revision.

2.1.2 Other Traditional Approaches

Other sequential analysis of documents concentrate on discrete spaces and discrete models, with the possible extension of [44]. Related papers on segmentation and sequential document analysis are [18, 2, 28] with [18] being the closest to our approach.

An influential model for topic modeling within and across documents is Latent Dirichlet allocation [4, 3]. Our approach differs in being fully non-parametric and in that it does not require iterative parametric estimation or integration. The interpretation of local word smoothing as a non-parametric statistical estimator [22] may be extended to our paper in a straightforward manner.

2.2 Text Visualizations

Several attempts have been made to visualize themes and topics in documents, either by keeping track of the word distribution or by dimensionality reduction techniques e.g.,[12, 16, 35, 39]. Such studies tend to visualize a corpus of unrelated documents as opposed to ordered collections of revisions which we explore.

Document visualization has gained considerable real world and research interest due to the inherent complexity of text and the overwhelming extent of digital text archives such as the Internet. Collections of version-controlled documents, such as code repositories and Google docs, compound these challenges by storing documents as they evolve over time and by several authors. Techniques for visualizing version-controlled documents tend to focus more on temporal and collaborative aspects and less on content.

A partial list of references for text visualization are [35, 15, 16, 12, 41, 3] with additional references available in [38]. A selection of software systems for visualizing text corpora are `http://in-spire.pnl.gov` (IN-SPIRE), `http://jheer.org/enron` (Enron corpus viewer), `http://www.refviz.com` (Thomson's refviz), and the `http://www.cs.cmu.edu/~lemur/science/` (Science topic browser).

2.2.1 Visualizing Word Histograms

Visualizing numeric data, such as word histograms, serves a foundational role in visualizing complicated textual objects. Monographs describing traditional visualization techniques are [5, 40] while less traditional approaches for visual data exploration are surveyed in [6]. Some interesting ideas concerning visualizing low-dimensional numeric time series are [45, 19]. Recent trends in the area of time series visualization are mostly concerned with interactive visualization and with multiple or vector-valued time series. An interesting exposition of the state-of-the-art and future vision in the related field of visual analytics is [38].

The use of n-grams to convert categorical sequences to numeric vectors is used extensively in the fields of information retrieval, speech recognition, and natural language processing. Recent monographs describing the use of n-grams in these areas are [20, 26, 1]. Visualizing n-grams is usually accomplished through statistical dimensionality reduction techniques. Methods such as principal component analysis and multidimensional scaling are surveyed in [11] while [23] reviews non-linear techniques for dimensionality reduction.

2.2.2 Sequential Visualization of Documents

Most of the methods mentioned above as well as other ones mentioned in [38] are designed for non-sequential visualization of a corpus of documents. Such methods only consider some measure of similarity between individual documents and as a result they are useful for browsing vast textual archives and nicely augment automatic search methods.

An alternative approach is the locally weighted bag-of-words representation that encodes documents as a series of local term-counts thereby allowing the visualization of sequential semantic progression within a single document [27, 22]. Other related work concerning sequential analysis of a single document includes TextTiling [17, 18] which partitions the data into multi-paragraph segments based on the local word histogram. Similarity scores between local word histograms at different document locations are used to segment the document into "text tiles" which were found to correspond well to subtopic boundaries according to human judgment. Similar ideas are explored in [33, 34, 46], in the statistical text segmentation literature e.g. [2] and in the text summarization literature e.g. [37].

Multi-resolution analysis provides a convenient mechanism for the sequential visualization of documents at several levels of granularity. The modern multi-resolution

analysis of data is inspired by the short time Fourier transform and wavelet representation [25]. An interesting application of multi-resolution wavelet analysis to document browsing is the Topic-Islands technique [29]. Topic-Islands constructs a digital signal that corresponds to the text document and then proceeds to compute its discrete wavelet transform. Multi-resolution visualization is then carried out by visualizing the energy of the various wavelet coefficients.

Our work is closest to the locally weighted bag of words approach mentioned above. In this approach categorical sequences are represented as smooth curves in the histogram space with document resolution tunable by varying the degree of smoothing. It generalizes two extremes–the original categorical sequence $y = \langle y_1, \ldots, y_N \rangle$ and the word histogram $\gamma^{\text{hist}}(y)$. By interpolating between these cases, it provides the flexibility of viewing sequential information at desired resolutions in a simple and effective manner.

The smoothness of the representation enables the use of tools from differential geometry and smooth analysis such as gradient, curvature, differential operators, and phase diagrams. These tools are used to create new visualization techniques for sequential trends in documents. Another interesting aspect of the proposed framework is that it draws interesting connections between document visualization and the considerable literature of local fitting [24] and functional data analysis [32] in statistics.

2.2.3 Visualizing Version-controlled Documents

Visualizations for version-controlled documents primarily focus on programming (code) repositories rather than more traditional documents. Although traditional document authoring is fundamentally different in style and content, one could imagine using these techniques for depicting the life-cycle of general documents.

History flow and Text-animated-transition [42, 10] are an outstanding visualization technique focusing on a changes in one document while most of visualization

technique is visualizing the history of entire repository. It shows the editing history of a document with vertical and horizontal flow(History flow) or an animation(Text-animated-transition).

Good overviews of techniques for visualizing the software evolution process are [36, 7]. Specific examples include SeeSoft [9], a line-by-line visualization of source code, as well as Augur[13] and Advizor[8]. The latter two are collections of visualizations, such as 2D and 2.5D matrix views which identify file and source changes in terms of project branch, date, author, etc. `http://fisheye.cenqua.com/`Cenqua Fisheye is one such commercial tool for visually interacting with software repositories, however the interface is text-centric and graphical displays consist of line charts and histograms. Perhaps more visually appealing, the StarGate project [30] and `http://social.cs.uiuc.edu/projects/codesaw.html`code saw serve roles similar to FishEye, i.e. tracking where and to what extent authors are concentrating their efforts, but provide a less static presentation.

An increasing number of recent visualizations emphasize aesthetics and result in a more qualitative rather than quantitative presentations of information. These so-called "organic visualizations" [14] use non-standard visual mechanisms, such as swirling clouds and blooming flowers, in which data members interact to exhibit emergent structure. These visualizations are not intended to replace traditional techniques, but serve a more complementary role. Notable examples include `http://code.google.com/p/gource/`gource and code_swarm [31].

These visualization tools are primarily intended to provide an understanding of the evolution of a *collection of documents*. Conversely, we present techniques for visually exploring the life-cycle of *one document*. Obviously both serve fundamentally different roles and answer different questions. As an example, visualizing a collection of documents could be more useful to a supervisor while visualizing the changes of

chapter would be of greater benefit to co-authors of a book. In addition to this fundamental difference, we hope to show that concrete mathematical concepts (such as norms of gradients) can be used to render interpretable and meaningful visualizations that are aesthetically pleasing.

CHAPTER III

LOCAL SPACE-TIME SMOOTHING

3.1 Space-Time Smoothing for Version-controlled Documents

With no loss of generality we identify the vocabulary V with positive integers $\{1, \ldots, V\}$ and represent a word $w \in V$ by a unit vector[1] (all zero except for 1 at the w-component)

$$e(w) = (0, \ldots, 0, 1, 0, \ldots, 0)^\top \quad w \in V. \tag{2}$$

We extend this definition to word sequences thus representing documents $\langle w_1, \ldots, w_N \rangle$ ($w_i \in V$) as sequences of V-dimensional vectors $\langle e(w_1), \ldots, e(w_N) \rangle$. Similarly, a version-controlled document is sequence of documents $d^{(1)}, \ldots, d^{(l)}$ of potentially different lengths $d^{(j)} = \langle w_1^{(j)}, \ldots, w_{N(j)}^{(j)} \rangle$. Using (2) we represent a version-controlled document as the array, where columns and rows correspond to space (document position) and time (versions).

$$\begin{array}{ccc} e(w_1^{(1)}), & \ldots, & e(w_{N(1)}^{(1)}) \\ \vdots & \ddots & \vdots \\ e(w_1^{(l)}), & \ldots, & e(w_{N(l)}^{(l)}) \end{array} \tag{3}$$

The array (3) of high dimensional vectors represents the version-controlled document without any loss of information. Nevertheless the high dimensionality of V suggests we smooth the vectors in (3) with neighboring vectors in order to better

[1]Note the slight abuse of notation as V represents both a set of words and an integer $V = \{1, \ldots, V\}$ with $V = |V|$.

capture the local word content. Specifically we convolve each component of (3) with a 2-D smoothing kernel K_h to obtain a smooth vector field γ over space-time [43] e.g.,

$$\gamma(s,t) = \sum_{s'} \sum_{t'} K_h(s-s', t-t') e(w_{s'}^{(t')}), \quad K_h(x,y) \propto \exp\left(-(x^2+y^2)/(2h^2)\right) \quad (4)$$

Thus as (s,t) vary over a continuous domain $\Omega \subset \mathbb{R}^2$, $\gamma(s,t)$, which is a weighted combination of neighboring unit vectors, traces a continuous surface in $\mathbb{P}_V \subset \mathbb{R}^V$. Assuming that the kernel K_h is a normalized density it can be shown that $\gamma(s,t)$ is a non-negative normalized vector i.e., $\gamma(s,t) \in \mathbb{P}_V$ (see (1) for a definition of $\mathbb{P}_V$) measuring the local distribution of words around the space-time location (s,t). It thus extends the concept of lowbow (locally weighted bag of words) introduced in [22] from single documents to version-controlled documents.

One difficulty with the above scheme is that the document versions $d_1, \ldots, d_l$ may be of different lengths. We consider two ways to resolve this issue. The first pads shorter document versions with zero vectors as needed. We refer to the resulting representation γ as the non-normalized representation. The second approach normalizes all document versions to a common length, say $\prod_{j=1}^{l} N(j)$. That is each word in the first document is expanded into $\prod_{j \neq 1} N(j)$ words, each word in the second document is expanded into $\prod_{j \neq 2} N(j)$ words etc. We refer to the resulting representation γ as the normalized representation.

The non-normalized representation has the advantage of conveying absolute lengths. For example, it makes it possible to track how different portions of the document grow or shrink (in terms of number of words) with the version number. The normalized representation has the advantage of conveying lengths relative to the document length. For example, it makes it possible to track how different portions of the document grow or shrink with the version number relative to the total document length. In either case, the space-time domain Ω on which γ is defined (4) is a two dimensional rectangular domain $\Omega = [0, I] \times [0, J]$.

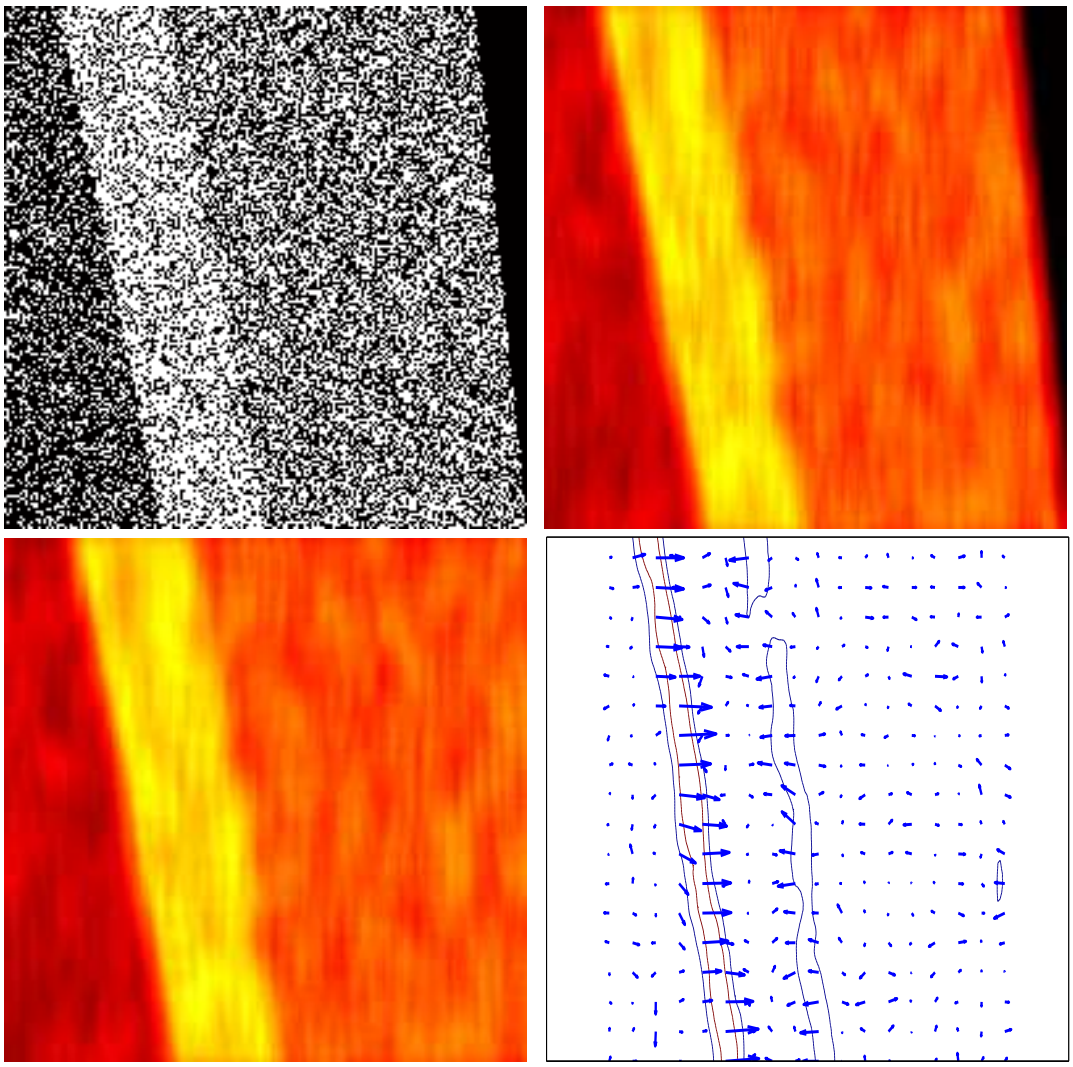

Figure 1: Four space-time representations of a simple synthetic version-controlled document over $V = \{1, 2\}$ (see text for more details). The top left panel displays the first component of (3) (non-smoothed array of unit vectors corresponding to words). The top right and bottom left panels display $[\gamma(s,t)]_1$ for the non-normalized and normalized representations respectively. The bottom right panel displays the gradient vector field $(\dot{\gamma}_s(s,t), \dot{\gamma}_t(s,t))$ (contour levels represent the gradient magnitude). The black portions of the first two panels correspond to zero padding due to unequal lengths of the different versions.

Before proceeding to examine how γ may be used, we demonstrate our framework with a simple low dimensional synthetic example. Assuming a vocabulary of two words $V = \{1, 2\}$, we can visualize γ by displaying its first component as a grayscale image (since $[\gamma(s,t)]_2 = 1 - [\gamma(s,t)]_1$ the second component is redundant). Specifically, we created a version-controlled document with three contiguous segments whose $\{1, 2\}$ words were sampled from Bernoulli distributions with parameters 0.3 (first segment), 0.7 (second segment), and 0.5 (third segment). That is, the probability of getting 1 is highest for the second segment, equal for the third and lowest for the first segment. The initial lengths of the segments were 30, 40 and 120 words with the first segment increasing and the third segment decreasing at half the rate of the first segment with each revision. The length of the second segment was constant across the different versions. Figure 1 displays the nonsmoothed ragged array (3) (top left), the non-normalized $[\gamma(s,t)]_1$ (top right) and the normalized $[\gamma(s,t)]_1$ (bottom left).

While the top left panel doesn't distinguish much, between the second and third segment the two smoothed representations display a nice segmentation of the space-time domain into three segments, each with roughly uniform values. The non-normalized representation (top right) makes it easy to see that the total length of the version-controlled document is increasing but it is not easy to judge what happens to the relative sizes of the three segments. The normalized representation (bottom left) makes it easy to see that the first segment increases in size, the second is constant, and the third decreases in size. It is also possible to notice that the growth rate of the first segment is higher than the decay rate of the third.

The bottom right panel of Figure 1 shows the gradient vector field corresponding to the synthetic version-controlled document described in the previous section. As expected, it tends to be orthogonal to the segment boundaries. Its magnitude is displayed by the contour lines which show highest magnitudes around segment boundaries.

3.2 Visualizing Change in Space-Time

We apply the space-time representation to four tasks. The first task, visualizing change, is described in this section. The remaining three tasks are described in the next three section.

The space-time domain Ω represents the union of all document versions and all document positions. Some parts of Ω are more homogeneous and some are less in terms of their local word distribution. Locations in Ω where the local word distribution substantially diverges from its neighbors correspond to sharp content transitions. On the other hand, locations whose word distribution is more or less constant correspond to slow content variation.

We distinguish between three different types of changes. The first occurs when the word content changes substantially between neighboring document positions within a certain document version. As an example consider a document location whose content shifts from high level introductory motivation to a detailed technical description. Such change is represented by

$$\|\dot{\gamma}_s(s,t)\|^2 = \sum_{w=1}^{V} \left(\frac{\partial [\gamma(s,t)]_w}{\partial s} \right)^2 . \tag{5}$$

A second type of change occurs when a certain document position undergoes substantial change in local word distribution across neighboring versions. An example is erroneous content in one version being heavily revised in the next version. Such change along the time axis corresponds to the magnitude of

$$\|\dot{\gamma}_t(s,t)\|^2 = \sum_{w=1}^{V} \left(\frac{\partial [\gamma(s,t)]_w}{\partial t} \right)^2 . \tag{6}$$

Expression (5) may be used to measure the instantaneous rate of change in the local word distribution. Alternatively, integrating (5) provides a global measure of

change

$$h(s) = \int \|\dot{\gamma}_s(s,t)\|^2 \, dt, \quad g(t) = \int \|\dot{\gamma}_t(s,t)\|^2 \, ds$$

with $h(s)$ describing the total amount of spatial change across all revisions and $g(t)$ describing the total amount of version change across different document positions. $h(s)$ may be used to detect document regions undergoing repeated substantial content revisions and $g(t)$ may be used to detect revisions in which substantial content has been modified across the entire document.

We conclude with the integrated directional derivative

$$\int_0^1 \|\dot{\alpha}_s(r)\dot{\gamma}_s(\alpha(r)) + \dot{\alpha}_t(r)\dot{\gamma}_t(\alpha(r))\|^2 \, dr \tag{7}$$

where $\alpha : [0,1] \to \Omega$ is a parameterized curve in the space-time and $\dot{\alpha}$ its tangent vector. Expression (7) may be used to measure change along a dynamically moving document anchor such as the boundary between two book chapters. The space coordinate of such anchor shifts with the version number (due to the addition and removal of content across versions) and so integrating the gradient across one of the two axis as in (6) is not appropriate. Defining $\alpha(r)$ to be a parameterized curve in space-time realizing the anchor positions $(s,t) \in \Omega$ across multiple revisions, (7) measures the amount of change at the anchor point.

Figure 2 shows the norm $\|\dot{\gamma}_s(s,t)\|^2$ (top left), $\|\dot{\gamma}_t(s,t)\|^2$ (top right) and the local maxima of $\|\dot{\gamma}_s(s,t)\|^2 + \|\dot{\gamma}_t(s,t)\|^2$ (bottom left) for a portion of the version-controlled Wikipedia Religion article. The first panel shows the amount of change in local word distribution within documents. High values correspond to boundaries between sections, topics or other document segments. The second panel shows the amount of change as one version is replaced with another. It shows which revisions change the word distributions substantially and which result in a relatively minor change. The third panel shows only the local maxima which correspond to edges between topics or segments (vertical lines) or revisions (horizontal lines).

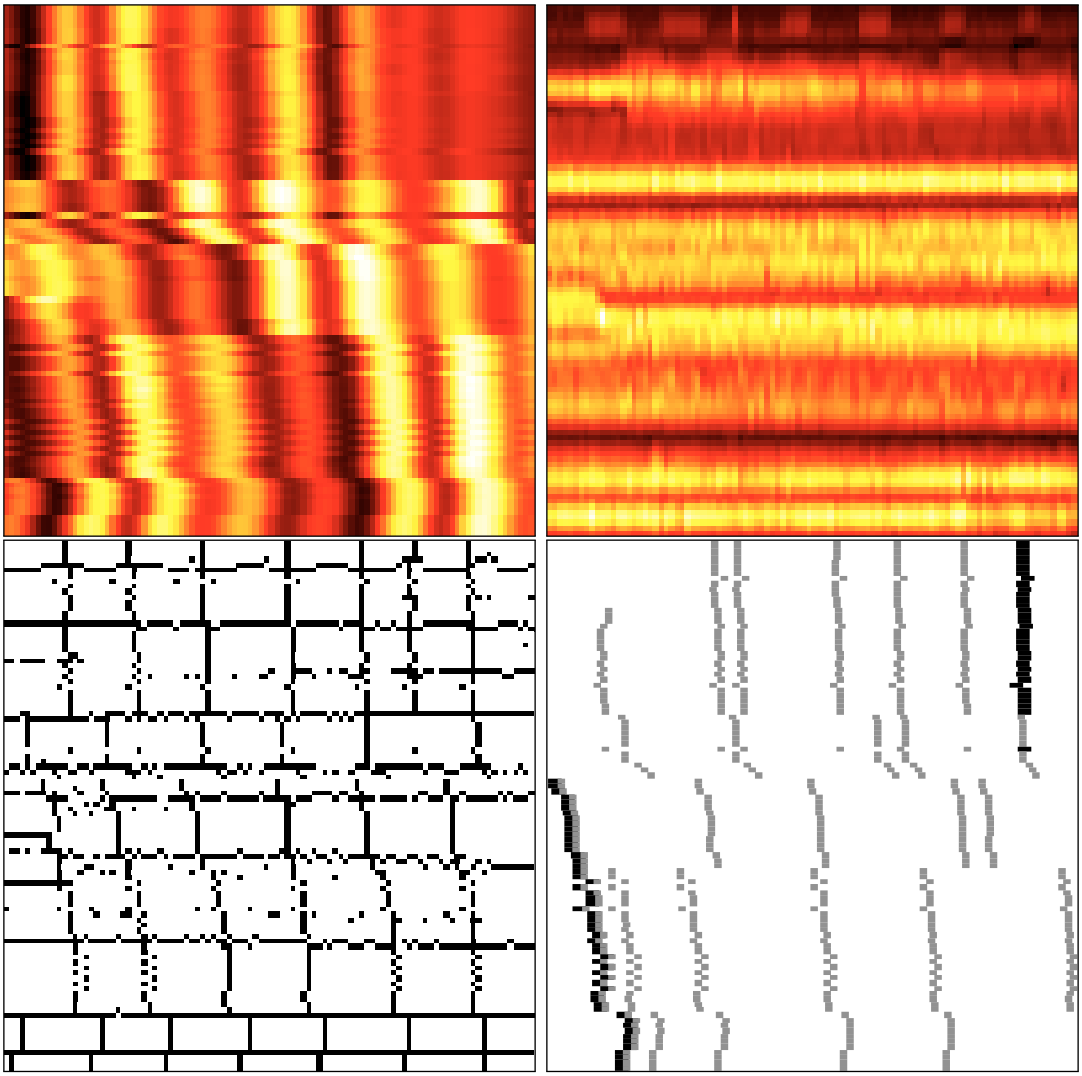

Figure 2: Gradient and edges for a portion of the version-controlled Wikipedia Religion article. The top left panel displays $\|\dot{\gamma}_s(s,t)\|^2$ (amount of change across document locations for different versions). The top right panel displays $\|\dot{\gamma}_t(s,t)\|^2$ (amount of change across versions for different document positions). The bottom left panel displays the local maxima of $\|\dot{\gamma}_s(s,t)\|^2 + \|\dot{\gamma}_t(s,t)\|^2$ which correspond to potential edges, either vertical lines (section and subsection boundaries) or horizontal lines (between substantial revisions). The bottom right panel displays boundaries of sections and subsections as black and gray lines respectively.

3.3 Edge Detection

In many cases documents may be divided to semantically coherent segments. Examples of text segments include individual news stories in streaming broadcast news transcription, sections in article or books, and individual messages in a discussion board or an email trail. For non-version-controlled documents finding the text segments is equivalent to finding the boundaries or edges between consecutive segments. See [18, 2, 28] for several recent studies in this area.

Things get a bit more complicated in the case of version-controlled documents. Segments, and their boundaries exist in each version. As in case of image processing, we may view segment boundaries as edges in the space-time domain Ω. These boundaries separate the segments from each other, much like borders separate countries in a two dimensional geographical map.

Assuming all edges are correctly identified, we can easily identify the segments as the interior points of the closed boundaries. In general, however, attempts to identify segment boundaries or edges will only be partially successful. As a result predicted edges in practice are not closed and do not lead to interior segments. We consider now the task of predicting segment boundaries or edges in Ω and postpone the task of predicting a segmentation to the next section.

Edges, or transitions between segments, correspond to abrupt changes in the local word distribution. We thus characterize them as points in Ω having high gradient value. In particular, we distinguish between vertical edges (transitions across document positions), horizontal edges (transitions across versions), and diagonal edges (transitions across both document position and version). These three types of edges may be diagnosed based on the magnitudes of $\dot{\gamma}_s$, $\dot{\gamma}_t$, and $\dot{\alpha}_1\gamma_s + \dot{\alpha}_2\gamma_t$ respectively.

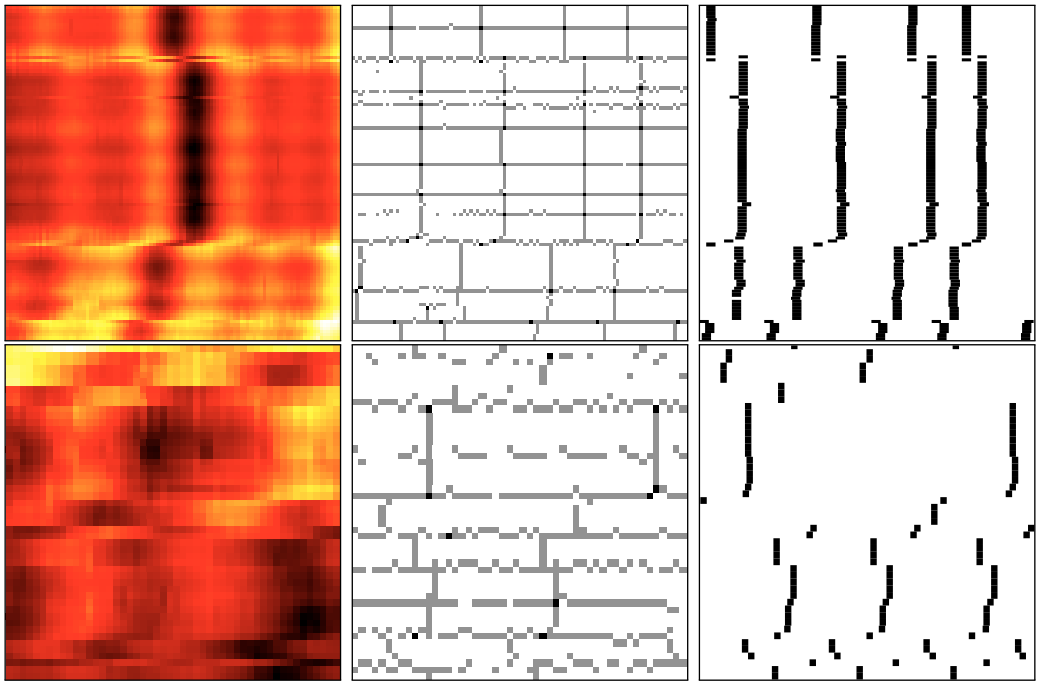

Figure 3: Gradient and edges of a portion of the version-controlled Atlanta Wikipedia article (top row) and the Google Wave Amazon Kindle FAQ (bottom row). The left column displays the magnitude of the gradient in both space and time $\|\dot{\gamma}_s(s,t)\|^2 + \|\dot{\gamma}_t(s,t)\|$. The middle column displays the local maxima of the gradient magnitude (left column). The right column displays the actual segment boundaries as vertical lines (section headings for Wikipedia and author change in Google Wave). The gradient maxima corresponding to vertical lines in the middle column matches nicely the Wikipedia section boundaries. The gradient maxima corresponding to horizontal lines in the middle column correspond nicely to major revisions indicated by a discontinuities in the location of the section boundaries.

Table 1: Test set error rate and F1 measure for edge prediction (section boundaries in Wikipedia articles and author change in Google Wave). The space-time domain Ω was divided to a grid with each cell labeled edge ($y = 1$) or no edge ($y = 0$) depending on whether it contained any edges. Method a corresponds to a predictor that always selects the majority class. Method b corresponds to the TextTiling test segmentation algorithm [18] without paragraph boundaries information. Method c corresponds to a logistic regression classifier whose feature set is composed of statistical summaries (mean, median, max, min) of $\dot{\gamma}_s(s,t)$ within the grid cell in question as well as neighboring cells.

Article	**Rev.**	**Voc. Size**	$p(y)$	**Error Rate**			**F1 Measure**		
				a	b	c	a	b	c
Atlanta	2000	3078	0.401	0.401	0.424	0.339	0.000	0.467	0.504
Religion	2000	2880	0.403	0.404	0.432	0.357	0.000	0.470	0.552
Language	2000	3727	0.292	0.292	0.450	0.298	0.000	0.379	0.091
European Union	2000	2382	0.534	0.467	0.544	0.435	0.696	0.397	0.663
Beijing	2000	3857	0.543	0.456	0.474	0.391	0.704	0.512	0.682
Amazon Kindle FAQ	100	573	0.339	0.338	0.522	0.313	0.000	0.436	0.558

Besides the synthetic data results in Figure 2, we conducted edge detection experiments on six different real world datasets. Five datasets are Wikipedia.com articles: Atlanta, Religion, Language, European Union, and Beijing. Religion and European Union are version-controlled documents with relatively frequent updates, while Atlanta, language, and Beijing have less frequent changes. The sixth dataset is the Google Wave Amazon Kindle FAQ which is a less structured version-controlled document.

Preprocessing included removing html tags and pictures, word stemming, stopword removal, and removing any non alphabetic characters (numbers and punctuations). The section heading information of Wikipedia and the information of author of each posting in Google Wave is used as ground truth for segment boundaries. This information was separated from the dataset and was used for training and evaluation (on testing set).

Figure 3 displays a gradient information, local maxima, and ground truth segment boundaries for the version-controlled Wikipedia articles Religion and Atlanta. The

local gradient maxima nicely match the segment boundaries which lead us to consider training a logistic regression classifier on a feature set composed of gradient value statistics (min, max, mean, median of $\|\dot{\gamma}_s(s,t)\|$ in the appropriate location as well as its neighbors (the space-time domain Ω was divided into a finite grid where each cell either contained an edge ($y = 1$) or did not ($y = 0$)).

The Table 1 displays the test set accuracy and F1 measure of three predictors: our logistic regression (method c) as well as two baselines: predicting edge/no-edge based on the marginal $p(y)$ distribution (method a) and TextTiling (method b) [18] which is a popular text segmentation algorithm. Since we do not assume paragraph information in our experiment we ignored this component and considered the document as a sequence with $w = 20$ and 29 minimum depth gaps parameters (see [18]). We conclude from the figure that the gradient information leads to better prediction than TextTiling (on both accuracy and F1 measure).

3.4 Segmentation

As mentioned in the previous section, predicting edges may not result in closed boundaries. It is possible to analyze the location and direction of the predicted edges and aggregate them into a sequence of closed boundaries surrounding the segments. We take a different approach and partition points in Ω to k distinct values or segments based on local word content and space-time proximity.

For two points $(s_1, t_2), (s_2, t_2) \in \Omega$ to be in the same segment we expect $\gamma(s_1, t_1)$ to be similar to $\gamma(s_2, t_2)$ and for (s_1, t_1) to be close to (s_2, t_2). The first condition asserts that the two locations discuss the same topic. The second condition asserts that the two locations are not too far from each other in the space time domain. More specifically, we propose to segment Ω by clustering its points based on the following geometry

$$d((s_1,t_1),(s_2,t_2)) = d_H(\gamma(s_1,t_1),\gamma(s_2,t_2)) + \sqrt{c_1(s_1-s_2)^2 + c_2(t_1-t_2)^2} \qquad (8)$$

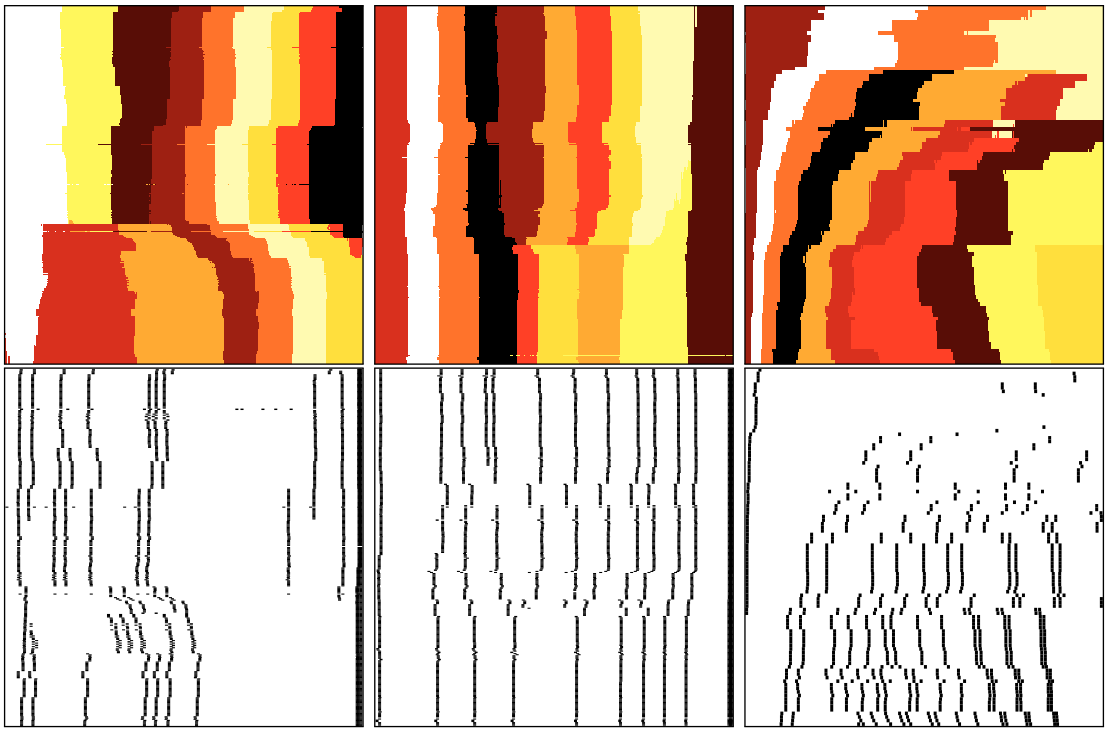

Figure 4: Predicted segmentation (top) and ground truth segment boundaries (bottom) of portions of the version-controlled Wikipedia articles Religion (left), Atlanta (middle) and the Google Wave Amazon Kindle FAQ(right). The predicted segments match the ground truth segment boundaries. Note that the first 100 revisions are used in Google Wave result. The proportion of the segments that appeared in the beginning is keep decreasing while the revisions increases and new segments appears.

where $d_H : \mathbb{P}_V \times \mathbb{P}_V \to \mathbb{R}$ is Hellinger distance

$$d_H^2(u, v) = \sum_{i=1}^{V} (\sqrt{u_i} - \sqrt{v_i})^2. \tag{9}$$

The weights c_1, c_2 are used to balance the contributions of word content similarity with the similarity in time and space.

Figure 4 displays the ground truth segment boundaries and the segmentation results obtained by applying k-means clustering ($k = 11$) to the metric (8). The figure shows that the predicted segments largely match actual edges in the documents even though no edge or gradient information was used in the segmentation process.

3.5 Predicting Future Operations

The fourth and final task is predicting a future revision d_{l+1} based on the smoothed representation of the present and past versions $d_1, \ldots, d_l$. In terms of Ω, this means predicting features associated with $\gamma(s,t), t \geq t'$ based on $\gamma(s,t), t < t'$.

We concentrate on predicting whether Wikipedia edits are reversed in the next revision. This action, marked by a label UNDO or REVERT in the Wikipedia API, is important for preventing content abuse or removing immature content (by predicting ahead of time suspicious revisions).

We predict whether a version will undergo UNDO in the next version using a logistic regression based on statistical summaries (mean, median, min, max) of the following feature set $\|\dot{\gamma}_s(s,t)\|$, $\|\ddot{\gamma}_s(s,t)\|$, $\|\dot{\gamma}_t(s,t)\|$), $\|\dot{\gamma}_t(s,t)\|$, $g(h)$, and $h(s)$. Table 2 shows the test set error and F1 measure for the logistic regression based on the smoothed space-time representation (method c), as well as two baselines. The first baseline (method a) predicts the majority class and the second baseline (method b) is a logistic regression based on the term frequency content of the current test version. Using the derivatives of γ, we obtain a prediction that is better than choosing majority class or logistic regression based on word content. We thus conclude that the derivatives above provide more useful information (resulting in lower error and higher F1) for predicting future operations than word content features.

Table 2: Error rate and F1 measure over held out test set of predicting future UNDO operation in Wikipedia articles. Method a corresponds to a predictor that always selects the majority class. Method b corresponds to a logistic regression based on the term frequency vector of the current version. Method c corresponds a logistic regression that uses summaries (mean, median, max, min) of $\|\dot{\gamma}_s(s,t)\|$, $\|\dot{\gamma}_s(s,t)\|$, $g(t)$, and $h(s)$.

Article	**Rev.**	**Voc. Size**	$p(y)$	**Error Rate**			**F1 Measure**		
				a	b	c	a	b	c
Atlanta	2000	3078	0.218	0.219	0.313	0.212	0.000	0.320	0.477
Religion	2000	2880	0.123	0.122	0.223	0.125	0.000	0.294	0.281
Language	2000	3727	0.189	0.189	0.259	0.187	0.000	0.334	0.455
European Union	2000	2382	0.213	0.208	0.331	0.209	0.000	0.275	0.410
Beijing	2000	3857	0.137	0.137	0.219	0.136	0.000	0.247	0.284

CHAPTER IV

CUMULATIVE REVISION MAP

4.1 Cumulative Revision Map

There is a traditional technique to track version-controlled documents, diff. The Unix diff solves Longest Common Subsequence problem, which finds the longest subsequence common to given two sequences. The information of diff gives which part was added or deleted in particular location. However, diff information is mainly for comparing two revisions. It is hard to track the change of a document with multiple revisions.

Cumulative Revision Map (CRM) augments the other visualization techniques mentioned above by displaying edits as rectangles in the space time matrix (rows correspond to revisions and columns to document position). Gray rectangles represent persistent edits-remaining until the final revision and red rectangles represent non-persistent-removed before the final version by a subsequent revision.

CRM maintains entire addition and deletion history of a document. The history is maintained by a graph structure with a node containing a subsequence of a document. In each revisions, CRM solves a longest common subsequence problem (like the unix diff command) between current sequence and the one has to be processed to find which part was added and deleted. CRM preserves unchanged part intact, split a node to add new content located in a node and to delete some parts of a node. In the process as shown figure 5, we can keep tracking of difference through revisions while preserving relevant position for each subsequences. Moreover, the vertical position of a node is determined by its revision.

The cumulative degree of change in a particular position could also be obtained

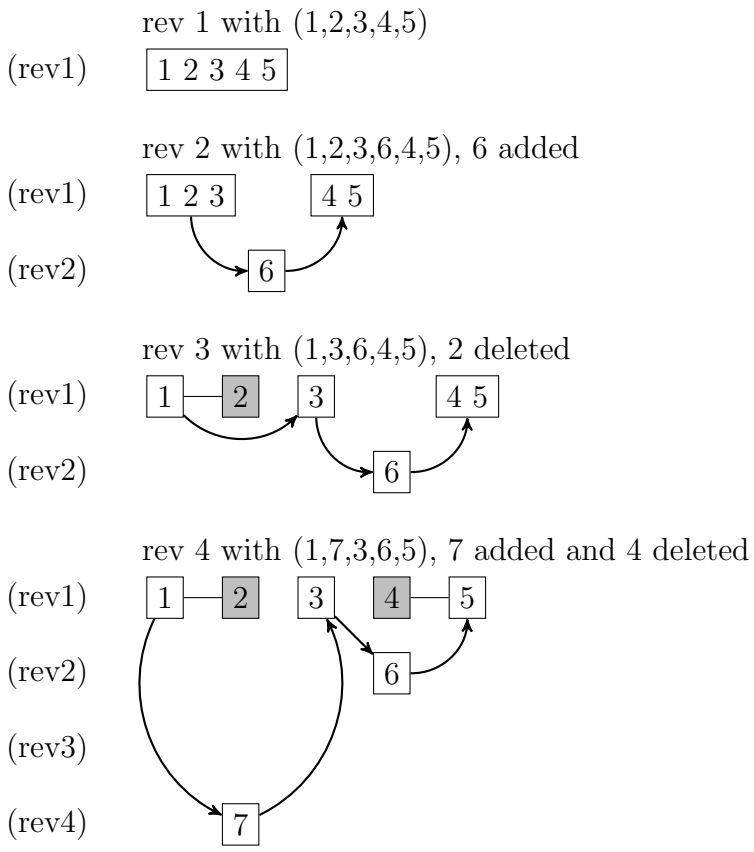

Figure 5: Cumulative Revision Map process. Started with a document (1,2,3,4,5) ((rev1). The node was split to insert a new content(rev2). Node '123' was split to mark the deleted node (rev3). Node '45' split and new node '7' was inserted.(rev4)

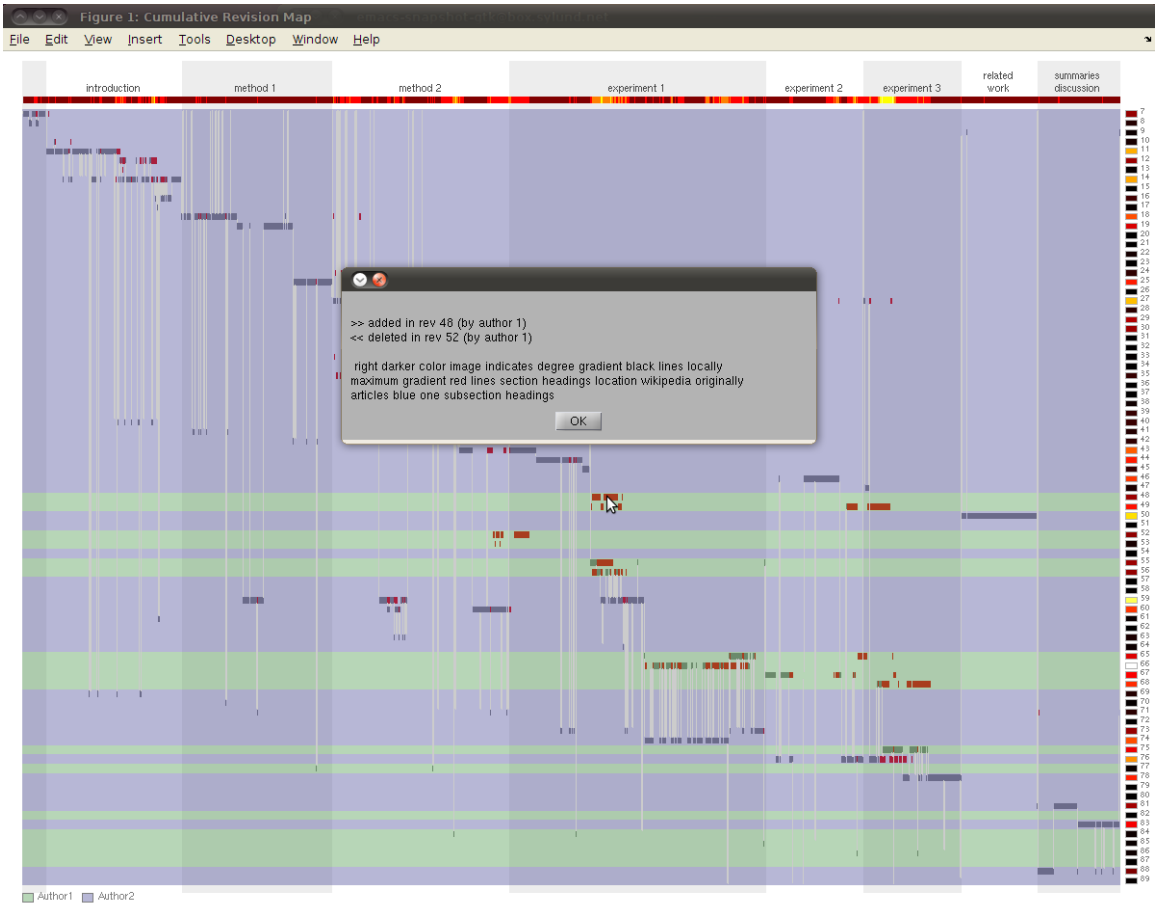

Figure 6: User-interaction of CRM. User can click on each nodes to obtain the information of the node.

from CRM. In figure 5 at revision 4, node 2 and 7 are located in the same relative position. It means the location between 1 and 3, 3 and 5 are changed two times while other positions are changed only 1 time. In a word, summing up all the number of changes in cumulative revision map along locations will be a degree of change in a particular document location. In a same way, the degree of change in a particular revision could also be computed with summing up changes along a revision.

The revision belongs to one user, so that vertical layout of CRM provides well understanding of who made a change. Generating unique colors for each author and also for major operation such as add or delete would gives an easy way to track an authoring process.

CRM is scalable and interactive. The nodes and edges could be simplified to give a clear representation of large datasets. Edges could be changed to vertical lines while shrinking all horizontal gaps between nodes. This simplification approach also gives more concise document location along horizontal layout. Moreover, user can pinpoint a node with mouse pointer to find out what was written and when the change was made (Figure 6).

4.2 Evaluation

We demonstrate our visualization techniques using several case studies using LaTeX document data, computer code, and Wikipedia articles. We focus on the following performance criteria: (a) how easy it is to determine what is the revision in which some change occurs, (b) how easy is it to track what part of the document is changed, (c) what was the content that was changed, and what was is the authorship pattern.

Figure 7 is a conference paper written in LaTeX by the authors of this paper. We see a striking diagonal pattern indicating sequential editing. In other words the authors worked their way from the beginning at early revisions toward the middle of the paper in middle revisions to the end of the paper in the final revisions. This sequential pattern is sometimes common but is by no means the only type of authorship pattern. In other cases different authors work in parallel on different parts of the paper (as in the programming code example of Figure 9. Another interesting pattern that we can extract from this Figure 7 is that the author 1 (green horizontal bands) authored a relatively little part of the document (around the middle) and is often being overruled by author 2 (purple horizontal bands). This is indicated by red color which corresponds to edits that are later removed or replaced with other content. Indeed author 2 corresponds to the adviser of author 1. Other interesting information is that the experiment section (middle part) had the most re-editing while the introduction, method, related work, and discussion were not rewritten many times.

Figure 7: Cumulative Revision Map of a conference paper. Document locations flow from left to right, and revisions does top to bottom. Gray boxes represents contents in use in latest revision, and red means deleted contents. Lines are connecting gray boxes along with the content of latest revision. Vertical background bar represents section and horizontal backgrounds shows authors with unique color. Top horizontal bar with spectrum shows degree of cumulative changes in document location: brighter color with higher change. Vertical segmented bar on the right side of each figure means cumulative change in a revision: brighter colors with higher changes.

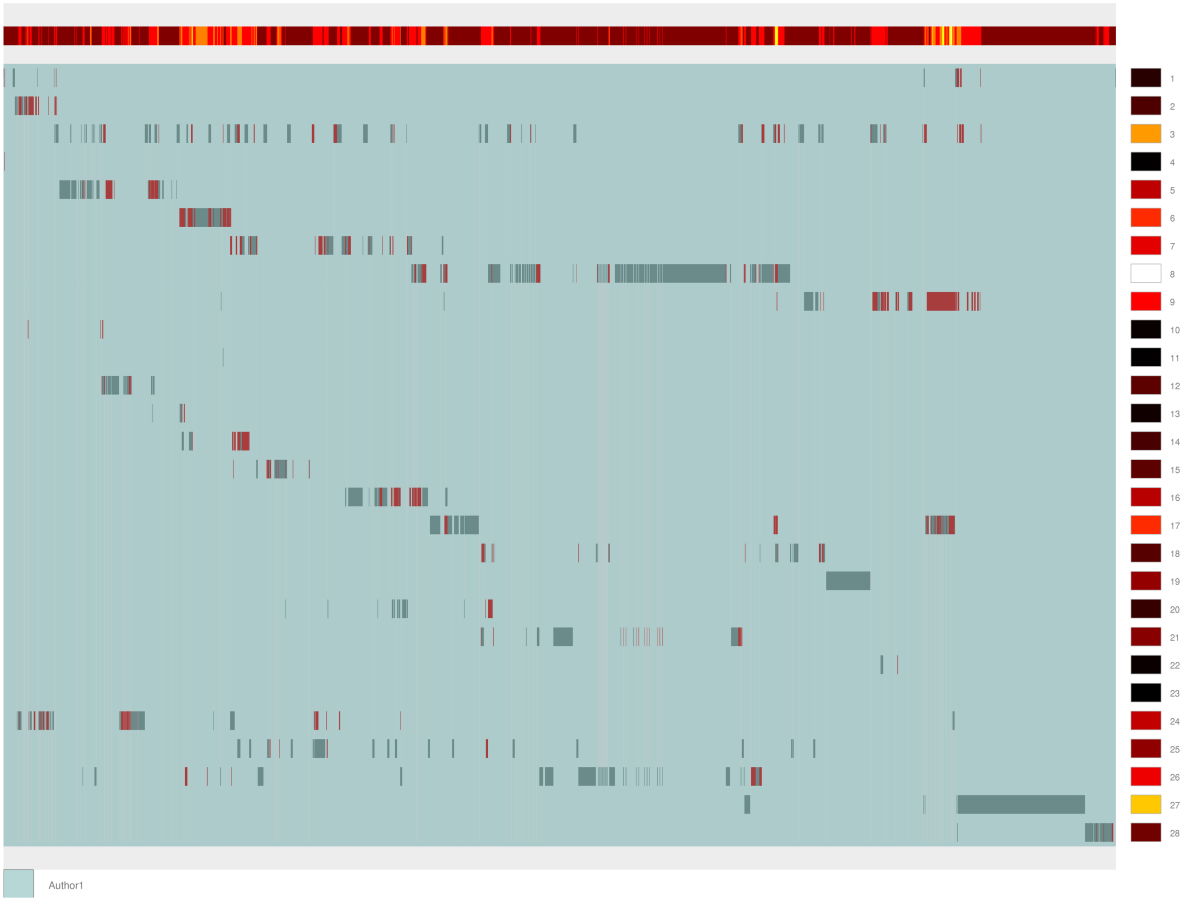

Figure 8: Cumulative Revision Map of a proposal slide. Document locations flow from left to right, and revisions does top to bottom. Gray boxes represents contents in use in latest revision, and red means deleted contents. Lines are connecting gray boxes along with the content of latest revision. Vertical background bar represents section and horizontal backgrounds shows authors with unique color. Top horizontal bar with spectrum shows degree of cumulative changes in document location: brighter color with higher change. Vertical segmented bar on the right side of each figure means cumulative change in a revision: brighter colors with higher changes.

Figure 8 shows another interesting authoring pattern. In it, we see multiple (four) sequential passes over the document from the beginning to the end. In each pass some content is changed in a neat sequential pass. The first pass contained relatively light editing (very rough draft) while the other three passes contained more edits.

Figure 9 contains computer code that is being dramatically overhauled. The many red rectangles represent non-persistent changes (edits that do not remain all the way to the final version). Indeed, it seems that the entire first 28 revisions were completely rewritten in the next 20 revisions. The lack of activity in the beginning of the document (left part does not contain gray or red rectangles) correspond to documentation that is left unchanged. This computer code shows a large deviation from the patterns described in the previous two cases. Computer code, in contrast to LaTeX document is not edited sequentially. Furthermore in the computer code example we had a large number of authors each working on a separate part of the code.

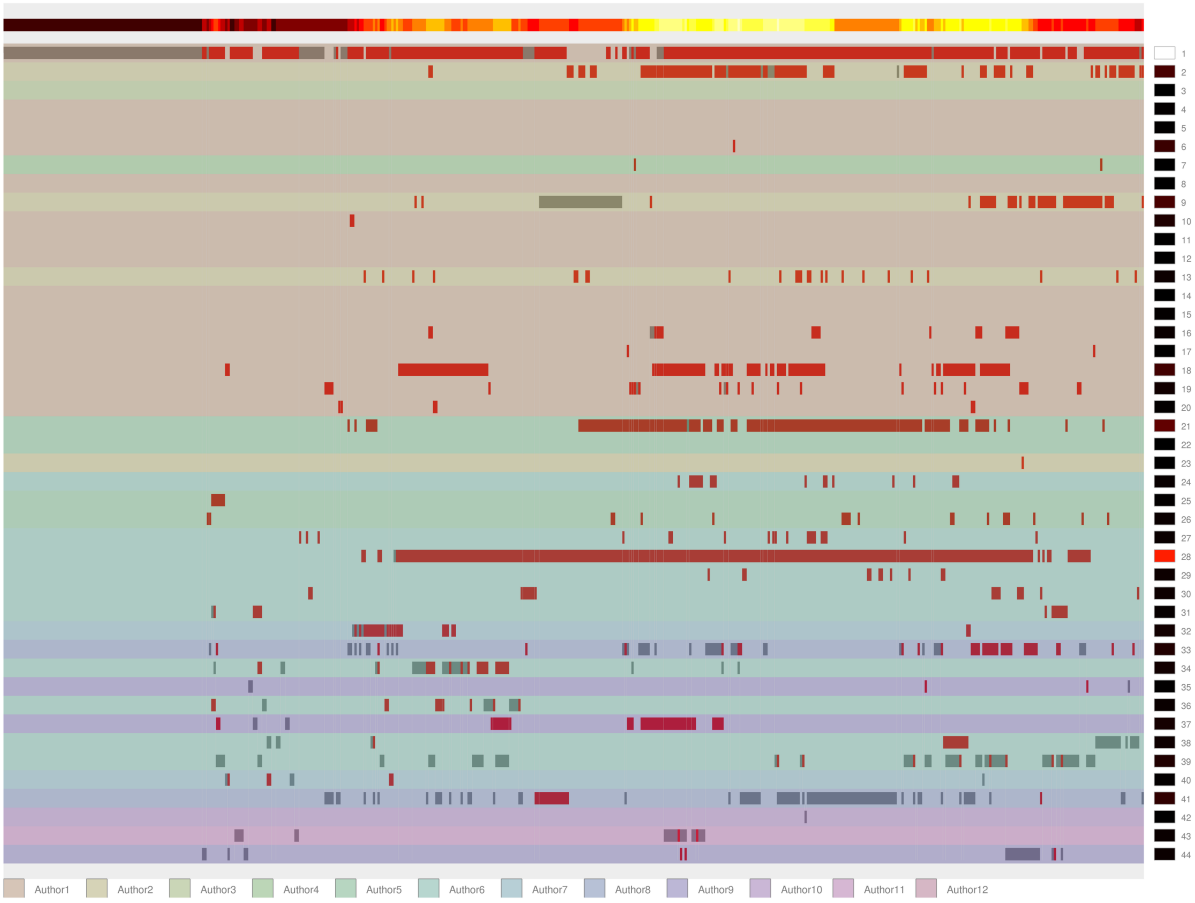

Figure 9: Cumulative Revision Map of a Google Tool Kit Compiler source file. Document location flows from left to right, and revisions does top to bottom. Gray boxes represents contents in use in latest revision, and red means deleted contents. Lines are connecting gray boxes along with the content of latest revision. Vertical background bar represents section and horizontal backgrounds shows authors with unique color. Top horizontal bar with spectrum shows degree of cumulative changes in document location: brighter color with higher change. Vertical segmented bar on the right side of each figure means cumulative change in a revision: brighter colors with higher changes.

CHAPTER V

DISCUSSION

The task of analyzing and visualizing version-controlled document is an important one. It allows external control and monitoring of collaboratively authored resources such as Wikipedia, Google Wave, and CVS or SVN documents. Our framework is the first to develop analysis and visualization tools in this setting. It presents a new representation for version-controlled documents that uses local smoothing to map a space-time domain Ω to the simplex of tf vectors $\mathbb{P}_V$. We demonstrate the applicability of the representation for four tasks: visualizing change, predicting edges, segmentation, and predicting future revision operations.

Visualizing changes may highlight significant structural changes for the benefit of users and help the collaborative authoring process. Improved edge prediction and text segmentation may assist in discovering structural or semantic changes and their evolution with the authoring process. Predicting future operation may assist authors as well as prevent abuse in coauthoring projects such as Wikipedia.

The experiments described in this thesis were conducted on synthetic, Wikipedia, Google Wave, and many SVN documents. They show that the proposed formalism achieves good performance both qualitatively and quantitatively as compared to standard baseline algorithms.

It is intriguing to consider the similarity between our representation and image processing. Predicting segment boundaries are similar to edge detection in images. Segmenting version-controlled documents may be reduced to image segmentation. Predicting future operations is similar to completing image parts based on the remaining pixels and a statistical model. Due to its long and successful history, image

processing is a good candidate for providing useful tools for version-controlled document analysis. Our framework facilitates this analogy and we believe is likely to result in novel models and analysis tools inspired by current image processing paradigms. A few potential examples are wavelet filtering, image compression, and statistical models such as Markov random fields.

The CRM technique complements the smoothing with discrete map displaying what content was edited in which revision and whether the content was persistent or not (overwritten or deleted in a future revision). This adds information concerning what document positions were edited in different times and by which author.

REFERENCES

[1] Baeza-Yates, R. and Ribeiro-Neto, B., *Modern Information Retrieval*. Addison Wesley, 1999.

[2] Beeferman, D., Berger, A., and Lafferty, J. D., "Statistical models for text segmentation," *Machine Learning*, vol. 34, no. 1-3, pp. 177–210, 1999.

[3] Blei, D. and Lafferty, J., "Dynamic topic models," in *Proc. of the International Conference on Machine Learning*, 2006.

[4] Blei, D., Ng, A., and Jordan, M., "Latent dirichlet allocation," *Journal of Machine Learning Research*, vol. 3, pp. 993–1022, 2003.

[5] Cleveland, W. S., *Visualizing Data*. Wadsworth Advanced Books and Software, 1993.

[6] de Oliveira, M. F. and Levkowitz, H., "From visual data exploration to visual data mining: A survey," *IEEE Transactions on Visualization and Computer Graphics*, vol. 9, no. 3, 2003.

[7] Diehl, S., *Software Visualization: Visualizing the Structure, Behaviour, and Evolution of Software*. Springer, 2007.

[8] Eick, S. G., Graves, T. L., Karr, A. F., Mockus, A., and Schuster, P., "Visualizing software changes," *Transactions on Software Engineering*, vol. 28, no. 4, pp. 396–412, 2002.

[9] Eick, S. G., Steffen, J. L., and E. E. Sumner, J., "Seesoft–a tool for visualizing line oriented software statistics," *Transactions on Software Engineering*, vol. 18, no. 11, pp. 957–968, 1992.

[10] F. Chevalier, P. Dragicevic, A. B. and Fekete, J., "Using text animated transitions to support navigation in document histories," in *Proceedings of the 28th international conference on Human factors in computing systems*, CHI '10, pp. 683–692, ACM, 2010.

[11] Fodor, I. K., "A survey of dimension reduction techniques," Technical Report UCRL-ID-148494, Lawrence Livermore National Laboratory, 2002.

[12] Fortuna, B., Grobelnik, M., and Mladenic, D., "Visualization of text document corpus," *Informatica*, vol. 29, pp. 497–502, 2005.

[13] Froehlich, J. and Dourish, P., "Unifying artifacts and activities in a visual tool for distributed software development teams," *International Conference on Software Engineering*, pp. 387–396, 2004.

[14] Fry, B. J., "Organic information design," Master's thesis, School of Architecture and Planning, MIT, 2000.

[15] Havre, S., Hetzler, E., Perrine, K., Jurrus, E., and Miller, N., "Interactive visualization of multiple query results," in *IEEE Symposium on Information Visualization*, pp. 105–112, 2001.

[16] Havre, S., Hetzler, E., Whitney, P., and Nowell, L., "Themeriver: Visualizing thematic changes in large document collections," *IEEE Transactions on Visualization and Computer Graphics*, vol. 8, no. 1, pp. 9–20, 2002.

[17] Hearst, M. A., "Multi-paragraph segmentation of expository text," in *Association of Computational Linguistics*, pp. 9–16, 1994.

[18] Hearst, M. A., "Texttiling: Segmenting text into multi-paragraph subtopic passages," *Computational Linguistics*, vol. 23, no. 1, pp. 33–64, 1997.

[19] Hochheiser, H. and Shneiderman, B., "Dynamic query tools for time series data sets, timebox widgets for interactive exploration," *Information Visualization*, vol. 3, no. 1, pp. 1–18, 2004.

[20] Jelinek, F., *Statistical methods for speech recognition*. MIT press, 1999.

[21] Kim, S. and Lebanon, G., "Local space-time smoothing for version controlled documents," in *Proc. of The 23rd International Conference on Computational Linguistics (COLING)*, 2010 (acceptance rate 42%).

[22] Lebanon, G., Mao, Y., and Dillon, J., "The locally weighted bag of words framework for documents," *Journal of Machine Learning Research*, vol. 8, pp. 2405–2441, October 2007.

[23] Lee, J. and Verleysen, M., *Nonlinear dimensionality reduction*. Springer, 2007.

[24] Loader, C., *Local Regression and Likelihood*. Springer, 1999.

[25] Mallat, S., *A Wavelet Tour of Signal Processing*. Academic Press, 1999.

[26] Manning, C. D. and Schutze, H., *Foundations of Statistical Natural Language Processing*. MIT Press, 1999.

[27] Mao, Y., Dillon, J., and Lebanon, G., "Sequential document visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 6, pp. 1208–1215, 2007.

[28] McCallum, A., Freitag, D., and Pereira, F., "Maximum entropy Markov models for information extraction and segmentation," in *Proc. 17th International Conference on Machine Learning*, pp. 591–598, 2000.

[29] Miller, N. E., Wong, P. C., Brewster, M., and Foote, H., "Topic islands - a wavelet based text visualization system," in *IEEE International Conference on Visualization*, pp. 189–196, 1998.

[30] Ogawa, M. and Ma, K.-L., "Stargate: A unified, interactive visualization of software projects," *In Proceedings of the 2008 IEEE Pacific Visualization Symposium*, pp. 191–198, 2008.

[31] Ogawa, M. and Ma, K.-L., "code swarm: A design study in organic software visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, no. 6, pp. 1097–1104, 2009.

[32] Ramsay, J. and Silverman, B. W., *Functional Data Analysis*. Springer, second ed., 2005.

[33] Salton, G., Allen, J., Buckley, C., and Singhal, A., "Automatic analysis, theme generation, and summarization of machine-readable texts," *Readings in Information Retrieval*, pp. 478–483, 1997.

[34] Salton, G., Singhal, A., Buckley, C., and Mitra, M., "Automatic text decomposition using text segments and text themes," in *UK Conference on Hypertext*, pp. 53–65, 1996.

[35] Spoerri, A., "InfoCrystal: A visual tool for information retrieval," in *Proceedings of the 4th Conference on Visualization*, 1993.

[36] Storey, M.-A. D., Cubrani, D., and German, D. M., "On the use of visualization to support awareness of human activities in software development: a survey and a framework," *SOFTVIS*, pp. 192–201, 2005.

[37] Teufel, S. and Moens, M., "Summarizing scientific articles: experiments with relevance and rhetorical status," *Computational Linguistics*, vol. 28, no. 4, pp. 409–445, 2002.

[38] Thomas, J. J. and Cook, K. A., eds., *Illuminating the Path*. IEEE Computer Society, 2005.

[39] Thomas, J. J. and Cook, K. A., eds., *Illuminating the Path: The Research and Development Agenda for Visual Analytics*. IEEE Computer Society, 2005.

[40] Tufte, E. R., *The Visual Display of Quantitative Information*. Graphic Press, 2 ed., 2001.

[41] Viégas, F. B., Golder, S., and Donath, J., "Visualizing email content: portraying relationships from conversational histories," in *Proc. of the conference on Human Factors in computing systems, CHI'06*, pp. 979–988, 2006.

[42] Viégas., F. B., Wattenberg, M., and Dave, K., "Studying cooperation and conflict between authors with history flow visualizations," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '04, pp. 575–582, ACM, 2004.

[43] Wand, M. P. and Jones, M. C., *Kernel Smoothing*. Chapman and Hall/CRC, 1995.

[44] Wang, C., Blei, D., and Heckerman, D., "Continuous time dynamic topic models," in *Proc. of Uncertainty in Artificial Intelligence*, 2009.

[45] Weber, M., Alexa, M., and Muller, W., "Visualizing time-series on spirals," in *Proc. of IEEE Symposium on Information Visualization*, pp. 7–14, 2001.

[46] Wong, P. C., Cowley, W., Foote, H., Jurrus, E., and Thomas, J., "Visualizing sequential patterns for text mining," in *Proc. of IEEE Symposium on Information Visualization*, pp. 1–5, 2000.